Name: Jude Waide User-name: jgnt42
Algorithm A: AS rank
Algorithm B: Genetic Algorithm
Description of enhancement of Algorithm A:

N.B. relevant comments in code are referenced by a number which can be searched for, e.g. in the code:
(004) this is a relevant comment
The main problem with ASrank is that it converges quickly and easily gets stuck in local minima. The algorithm gets stuck when pheromone levels run out except on the chosen path, meaning no more exploration of the search space is being done. To detect when stagnation occurs, the tour length of the top 80% ant tours are compared (001). If all of the tour lengths are the same, then it is reasonable to assume that all of these ants have the same tour (since it's unlikely that 2 different tours have the same length) and that stagnation has occurred. The 80% threshold can be adjusted through the variable 'stagnationThreshold'. If stagnation is detected, pheromone smoothing is performed (002). This readjusts all pheromone levels to be a fraction of the initial pheromone amount, with the exact fraction on an edge depending on how much pheromone was on that edge prior. The variable gamma can also be modified to increase how much pheromone is placed on dead edges, 0 means the pheromone levels are not changed, whilst 1 causes all pheromone levels to be reset to the initial pheromone amount. This means that information learnt of good paths is still retained, whilst also opening up the entirety of the search space for more exploration, preventing the algorithm from getting trapped in a local minima.
The 2nd enhancement encourages unexplored areas of the search space to be explored through the use of an originality matrix. This keeps track of how many times an edge has been traversed by an ant over the running of the algorithm (003). The top w ants with the shortest tours are then ranked by how original they are (the sum of the originality of every edge in the tour) (004). This ranking is then used for the ranked pheromone deposit, instead of just ranking by tour length as in standard ASrank.
Finally, the whole algorithm has been made to use multiprocessing - each iteration the creation of ant tours is distributed equally among a certain number of processes (005). This changes the structure of the code a bit, however the exact same operations are being executed at the basic algo, just faster. This means more of the search space can be explored in 1min.

Description of enhancement of Algorithm B:

The 1st enhancement is an updated crossover function (001). The crossover function in the basic implementation gets 2 random points, and swaps the subtour between these 2 points in both parents. Since the same 2 random points are used for both parents, both subtours are in the same position and have the same length. The updated crossover function uses 4 random points, results in different length subtours that are in different positions. This gives a wider range of possible children, resulting in more diversity in the gene pool and exploration of the search space.
The 2nd enhancement is an updated mutation function (002). In the basic implementation, exchange mutation is used in which 2 random cities in a tour are swapped around. Instead, inverted displacement mutation is used in which a subtour is selected by using 2 random pointers. The elements of this subtour are then inverted, before the whole subtour is moved to a random position in the full tour. This mutation is more powerful than exchange mutation, again leading to more diversity in the gene pool.
The 3rd enhancement is a new fitness function (003). Rather than just 1 over the tour length, the new function finds how far the tour length is between the best tour length found so far, and the largest tour length of the current population. This always outputs a value between 0 and 1. This creates more of a distinction between good tours and bad tours, making good tours more likely to be selected as parents. This increases how quickly the algorithm converges to a good answer.
At each iteration, the 4th enhancement (004) takes the 5 shortest unique tours from the old population and the 5 shortest unique tours from the new population. The 5 shortest of these 10 are then combined with the whole of the new population (excluding the top 5 by tour length to maintain the pop size, as well as culling duplicate genes). This maintains some diversity within the gene pool and also ensures good tours are not forgotten.
The 5th enhancement (005) then multiplies the fitness of these top 5 tours by 3 * (5-rank), so the better the rank, the more likely it is to be selected as a parent.
Finally, the whole process is made to use multiprocessing to split the creation of the new population evenly between multiple processes (006). This does not change any calculations done compared to the basic version, but only makes it quicker.

DESCRIPTION OF ALGORITHM ONLY IF THE ALGORITHM IS NOT COVERED IN LECTURES

Description of *non-standard* Algorithm A:

Describe any non-standard algorithms you have implemented that ***have not been covered in lectures*** *(otherwise these boxes should be blank) You need to convince me that your implementation is indeed that of the named algorithm and you need to* ***provide a full reference to the source for your algorithm****. You should* ***include a pseudocode description****. You can vary the sizes of these boxes but do not change the font (Calabri), font size (11), the paragraph properties (single space) or the header and footer and everything should fit onto one side of A4. (You can delete these instructions.)*

Remember: You need my express permission to implement a non-standard algorithm!

Description of *non-standard* Algorithm B:

Type here, as above.